ADVANCED TRANSACTION PROCESSING

Transaction Processing Monitor

- It was developed for **building complex transaction processing** systems with a **large number of clients and servers**.
- Transaction Processing Monitors act as **middleware** (middleware is software that helps and bridges a variety of communication/connectivity between two or more applications)
- Its main task is to support and handle interactions between applications on a variety of computer platforms.
- Transaction Processing Monitors also usually known as TP-monitors provide functionalities such as managing, deploying, and developing transactional distributed information systems. It controls programs that monitor or manage a transaction of data as it passes from one stage in a process to another in an organized transaction-oriented manner.

Transaction Processing Monitor

- Transaction processing monitor is a program that monitors transactions from one stage to the next.
- The main aim of transaction processing monitor is to ensure that the transaction is processed either completely or not at all.
- And if any transaction is processed partially, it would bring the database to its previous consistent state.
- If any error occurs in between, then it would take appropriate actions.
- It is used in 2-tier, 3-tier and n-tier architectures.
- It also takes care about resource sharing is done appropriately among applications.

Transaction Processing Monitor

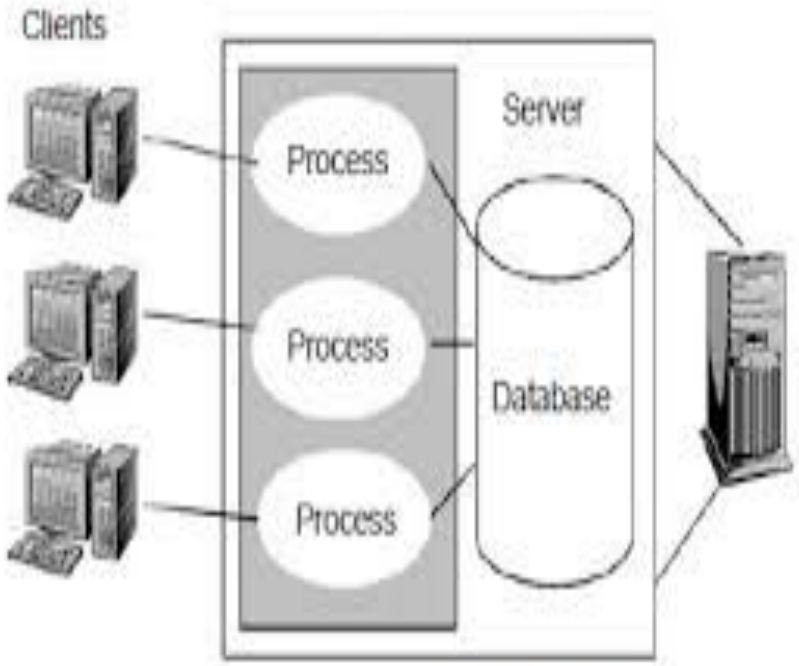
- It provides infrastructure for building and administering complex transaction processing systems with a large number of clients and multiple servers.

Duties carried out by TP Monitor:

- It provides the ease of creating user interfaces.
- It unwraps the incoming content/data into data packets.
- It provides a continuous row/queue of client requests and responses from the server.
- It routes the client data to servers.
- It gives secure returns to services.
- It hides inner transmission details from programmers.
- Helps in maintaining the load of the program.

Transaction Processing Monitor Architectures

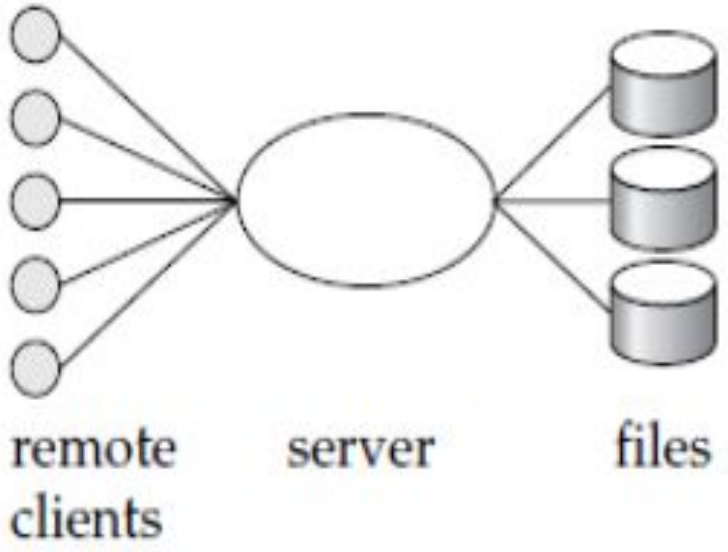
- **Process per client model:**
- In this model, each client would be handled by each server process, which authenticates the client and executes the action requested by the client.
- Problems occurring in this model are as follows:
- This model needs huge amount of memory.
- CPU overhead occurs as, OS divides up CPU time among processes by process of switching called multitasking.



Process-per-client Architecture

Transaction Processing Monitor Architectures

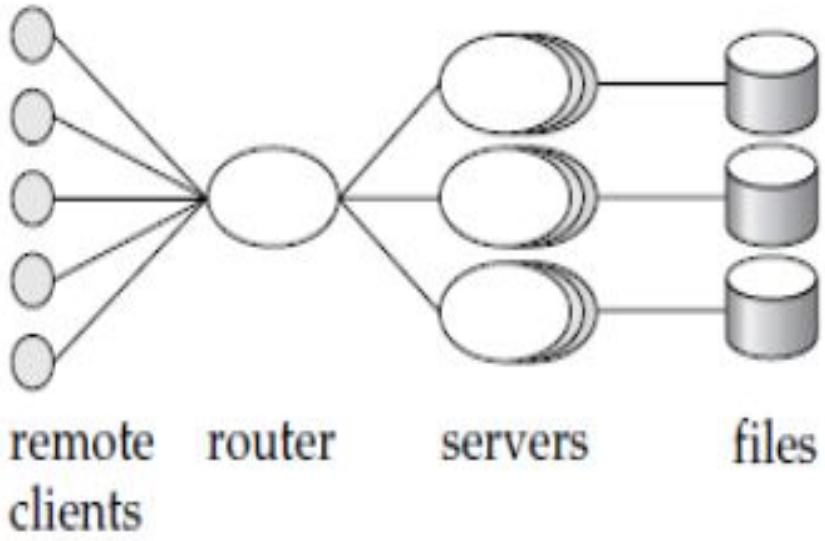
- **Single Server Model**
- In this, remote client sends the request to the server process which then executes those requests.
- The server processes handles tasks, such as user authentication, basically handled by operating systems.
- The server process is multithreaded, so that other clients do not get blocked while processing long requests arriving from each client. It helps in performing low overhead multitasking all together.
- Cost of switching between threads is low compared to previous model.
- This model overcomes the problems, which occur in process per client model.



Single-server model

Transaction Processing Monitor Architectures

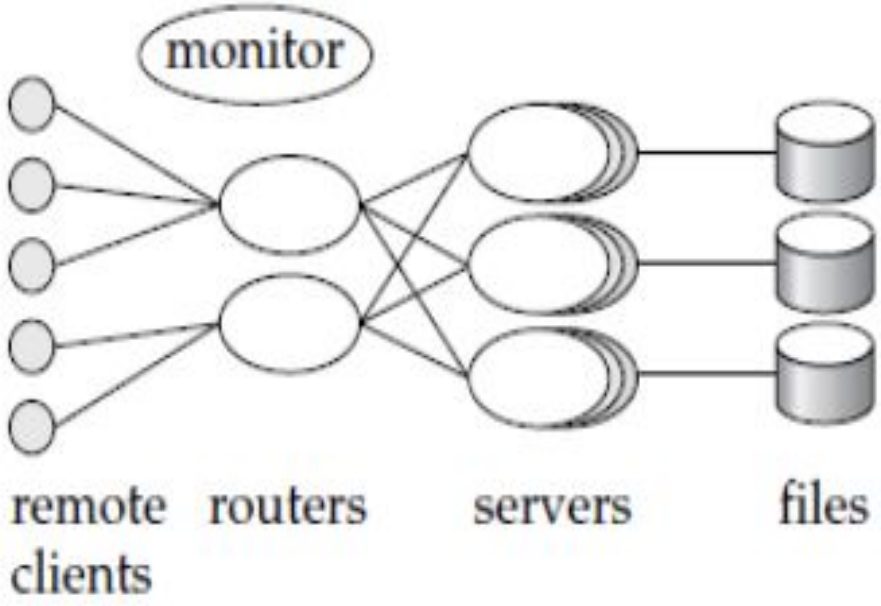
- **Many server, Single router**
- Multiple application server processes access a common database, clients communicate with the application through a single communication process that routes requests.
- It provides independent server processes for multiple applications
- Multithreaded server process
- Runs on parallel or distributed database servers



Many-server, single-router model

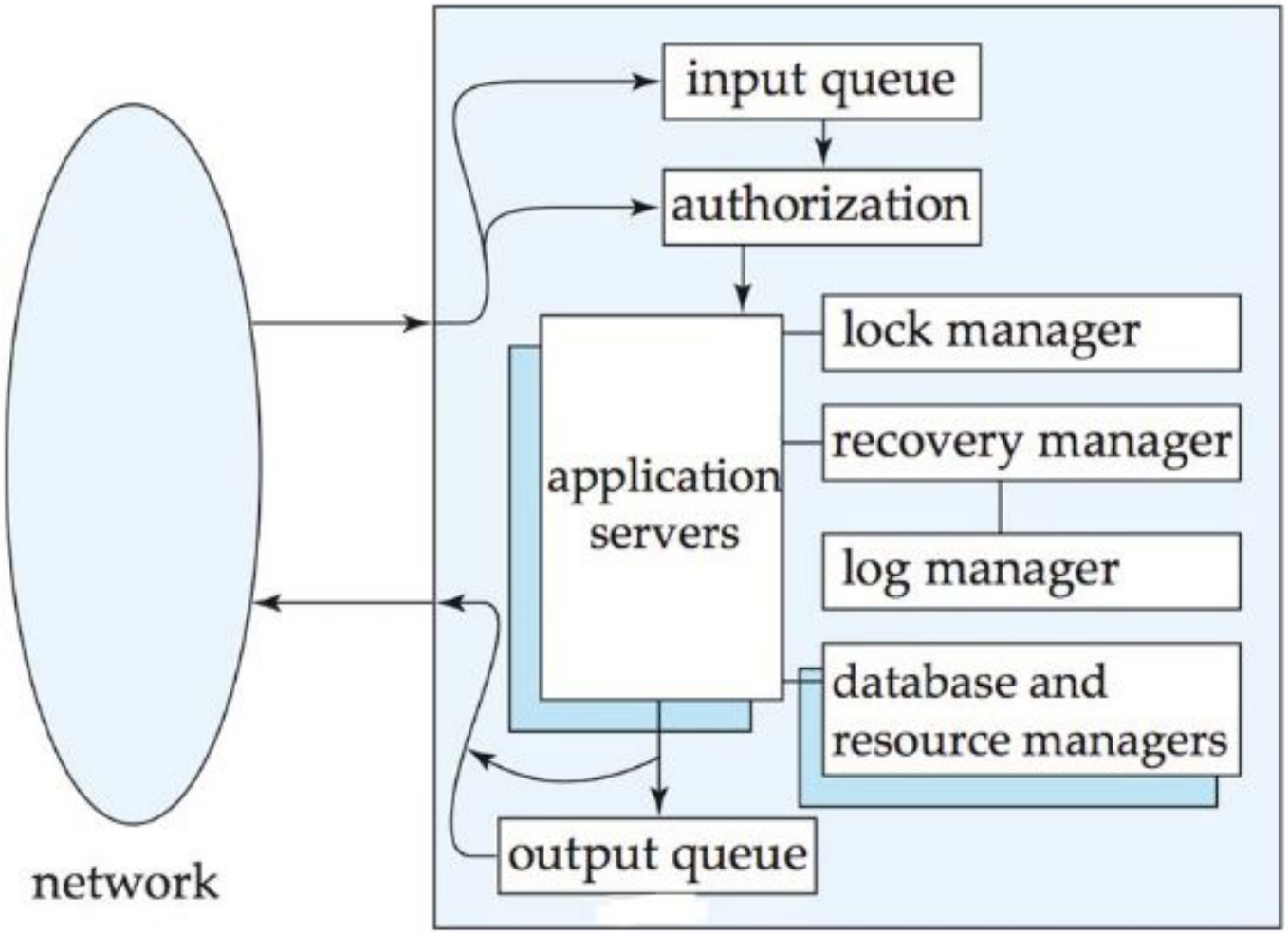
Transaction Processing Monitor Architectures

- **Many-server, many router model**
- Multiple processes communicate with clients.
- Client communication processes interact with router processes that route their request to the appropriate server.
- Controller process starts up and supervises other processes.



Many-server, many-router model

TP-Monitor Structure/Components



TP-Monitor Structure/Components(cont..)

- Queue manager manages the queue of incoming messages from clients.
- Queue manager provides durable or persistent queuing of messages.
- Regardless of system failure, messages stored in queue will be processed eventually.
- On transaction commit, TP monitor takes care of message is delivered to client regardless of system crash.
- ACID properties are provided for sending messages outside the database system.

TP-Monitor Structure/Components(cont..)

- TP monitors also provide locking, logging and recovery services, so that application servers can implement it by themselves.
- Interface between TP monitor and resource manager is defined by a set of transaction primitives.
- Resource manager interface is defined by X/Open distributed transaction processing standard
- TP monitor controls the scope of failure by administering server pools.
- TP monitor system provides a transactional remote procedure call interface to their services.

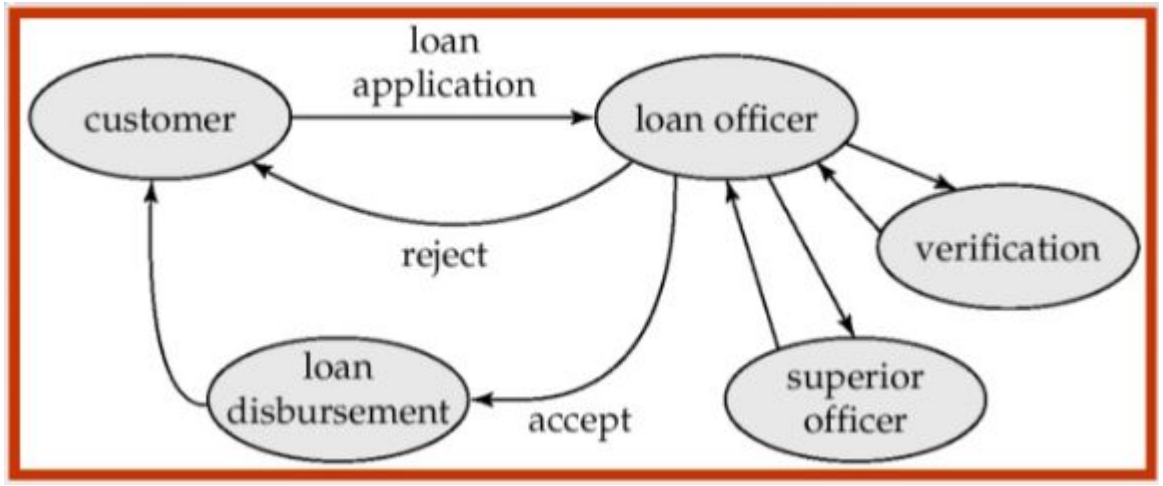
Transactional Workflow

- Workflows are activities that involve the coordinated execution of multiple tasks performed by different processing entities.
- The processing entity that performs the tasks may be a person or a software system(eg. A mailer, an application program or database management system)
- With the growth of networks, and the existence of multiple autonomous database systems, workflows provide a convenient way of carrying out tasks that involve multiple systems.
- Example: electronic mail routing.

Transactional Workflow Examples

Workflow application	Typical task	Typical processing entity
electronic-mail routing	electronic-mail message	mailers
loan processing	form processing	humans, application software
purchase-order processing	form processing	humans, application software, DBMSs

Loan Processing System



- Computerized workflow aim to automate many of the tasks.
- But still humans play an important role. E.g. in approving loans.
- The coordination of the task is carried out by passing the loan application with attached notes and other information from one employee to another in the form of electronic mail.

Aspects of Computerized/Automated Workflow

1. **Specification of workflows** – Details of the tasks to be carried out and defining its execution requirements.
2. **Workflow Execution** – Execute the transactions specified in the workflow while also provide traditional database safety related to the computations, data integrity and durability.

- For e.g. Loan application should not get lost even if system failure occurs.

Workflow specification

- The coordination of tasks can be specified either statically or dynamically.
- **Static Specification:**
- It defines the tasks and dependencies among them before the execution of the workflow begins.
- The dependencies among tasks may be simple each one needs to be completed before the next begins i.e. preconditions need to be satisfied.
- Pre – Conditions like:
- Execution states of other tasks.
- Output values of other tasks.
- External variables, that are modified by external events.

Workflow specification (cont..)

- **Dynamic task coordination**
- E.g. Electronic Mail Routing System in which the task to be scheduled for a given mail message depends on the destination address and on which intermediate routers are functioning.

Failure-Atomicity Requirements

- Using ACID transactional requirements is too strong/unimplementable for workflow applications.
- However, workflows must satisfy some limited transactional properties that guarantee a process is not left in an inconsistent state.
- **Acceptable termination states:**
- Every execution of a workflow will terminate in a state that satisfies the failure-atomicity requirements defined by the designer.
  - Commit
  - Abort
- A workflow must reach an acceptable termination state even in the presence of system failure.

Execution of Workflows

- The execution of the tasks may be controlled by a human coordinator or by a software system.
- A workflow management system consists of a scheduler, task agent, and mechanism to query the state of the workflow system.
- **Scheduler:** A program that processes workflows by submitting various tasks for execution, monitoring various events, and evaluating conditions related to inter-task dependencies. Responsible for ensuring tasks reach acceptable termination states.
- **Task agent:** Controls the execution of a task by a processing entity.

Thank You